Annotation	Description	Input Example	Output

@JsonIgnore	Ignores a field during serialization/deserialization.	public class User {
		public String name;
		@JsonIgnore
		public String password;
		}
			{"name": "John"}

@JsonInclude	Include only non-null/non-empty values in JSON.	@JsonInclude(JsonInclude.Include.NON_NULL)
		public class Product {
		public String name;
		public String description;
		}
			{"name": "Laptop"}

@JsonFormat	Format dates or other types during serialization.	public class Event { public LocalDate eventDate; } @JsonFormat(pattern = "yyyy-MM-dd")	{"eventDate": "2025-07-29"}

@JsonAlias	Accepts multiple names for a field during deserialization.	public class Car {
		@JsonAlias({"car_name", "vehicleName"})
		public String name;
		}
			{"car_name": "Toyota"} or {"vehicleName": "Toyota"}

@JsonAnyGetter / @JsonAnySetter	Handle unknown properties dynamically.	public class Config {
		private Map<String, String> settings = new HashMap<>();
		@JsonAnySetter
		public void set(String key, String value) {
		settings.put(key, value);
		}	

```
  }


  @JsonAnyGetter
  public Map<String, String> getSettings() {
    return settings;
  }
}
```

```
{"theme": "dark", "fontsize": "14"}
```

@JsonEnumDefaultValue	Assign default enum if value doesn't match any enum.

```
public enum Status {
  ACTIVE,
  INACTIVE,
```

```
  @JsonEnumDefaultValue

  UNKNOWN

}
```

If input is "PAUSED", deserialized as UNKNOWN

@JsonIgnoreProperties	Ignore specified fields from

	serialization/deserialization.	@JsonIgnoreProperties({"age", "email"}) public class Person { public String name;	{"name": "Alice"}

@JsonIncludeProperties	Only include specified fields during serialization.	@JsonIncludeProperties({"name", "email"})
		public class Person {
		public String name;
		public String email;
		public int age;
		}
			{"name": "Alice", "email": "alice@example.com"}

@JsonFilter	Apply filtering dynamically using filter ID.	@JsonFilter("userFilter")
		public class User {
		public String username;
		public int age;
		}
			Depends on applied filter logic

@JsonIdentityInfo	Handle cyclic references using object IDs.	@JsonIdentityInfo(generator = ObjectIdGenerators.PropertyGenerator.class, property = "id")
		public class Employee {
		public int id;
		public String name;
		public Department department;
		}
			{"id": 1, "name": "Sam", "department": 101}

@JsonIdentityReference	Serialize the referenced object as an ID instead of whole object.	public class Employee {
		public int id;
		public String name;
		@JsonIdentityReference(alwaysAsId = true)
		public Department department;
		}
			{"department": 101}

@JsonManagedReference / @JsonBackReference	Manage bidirectional relationships.	public class User {
		public String name;
		@JsonManagedReference
		public List<Address> addresses;
		}	

```
public class Address {
  public String city;
  @JsonBackReference
  public User user;
}
```

```
{"name": "John", "addresses": [{"city": "NY"}]}
```